WIA1002/WIB1002 DATA STRUCTURE

Mid Term Test (15%)

INSTRUCTION:

- **Create a Netbeans project entitled "YourMatrixNumber_MidTerm" and save your whole project to Z drive.**
- **Type your name, matrix number, tutorial group and tutorial lecturer's name on each file.**
- **Put away all your books/notes. Off your mobile phone. Do not communicate with anyone except your lecturer or teaching assistants. Treat this as a real exam.**
- **Any form of misconduct (copying) will be severely penalized. ZERO mark when you look at your phone, open any browser / chat program or try in any ways to communicate with others.**
- **Time allocation: 3 hours**
- **This test consists of 3 questions.**

Question 1 (5 marks)

A *ring* is a collection of items that has a reference/marker to a current item. An operation called *advance* moves the marker to the next item in the collection. When the marker reaches the last item, the next *advance* operation will move the marker back to the first item.

The following are the ring's operations:

i. add – adds an entry to the **end** of this ring
ii. remove – removes and returns the entry at the **beginning** of this ring
iii. getCurrentEntry – returns the current entry in this ring. Hint: The marker refers to the current entry in the ring.
iv. advance – advances the marker of the current entry in this ring by one entry.
v. getLength –gets the number of entries in this ring
vi. isEmpty – checks whether this ring is empty.

a) Based on the description above, design an ADT to represent a ring of objects. You do not have to draw a UML class diagram for the ADT but you have to write a Java interface named *RingInterface<T>* to specify the above operations of a ring. Specify each operation by stating its purpose, parameters and return value, using javadoc-style comments.

The *RingInterface* must be saved in a file named "RingInterface.java".

Question 2 (5 marks)

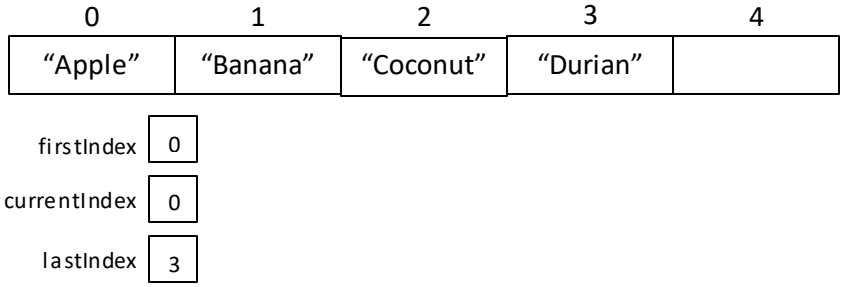
a) Define a class *ArrayRing* that implements your *RingInterface<T>* from Question 1. An object of *ArrayRing* can store maximum 5 entries only. Use an array in your implementation. The *ArrayRing* class must be saved in a file named "ArrayRing.java". No marks will be given for any implementation using the existing Collection classes from the Java API Library.

Use a fixed size array of size 5.
You must have the following data members:

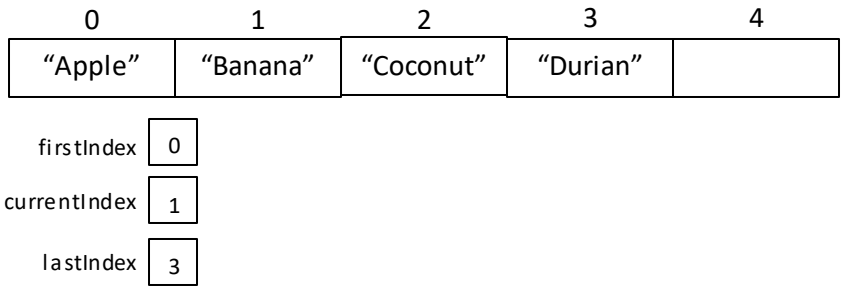
firstIndex - Marks the first entry
lastIndex - Marks the last entry
currentIndex - Marks the current entry. This is the reference/marker to the current item.

Initially *currentIndex* marks the first entry.
Calling the *advance* method will advance *currentIndex* by one entry.
Add method will add an item to the end of the ring but cannot add an item if there is no more empty slot in the ring.

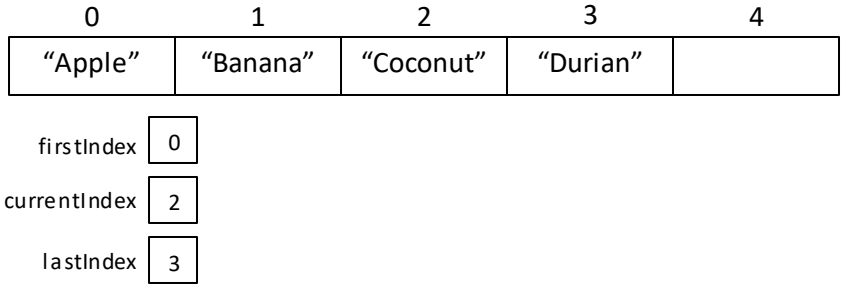
Read part b) especially the required output together with the following illustrations before you start this question. This will help you to get a better understanding.



After invoking advance():

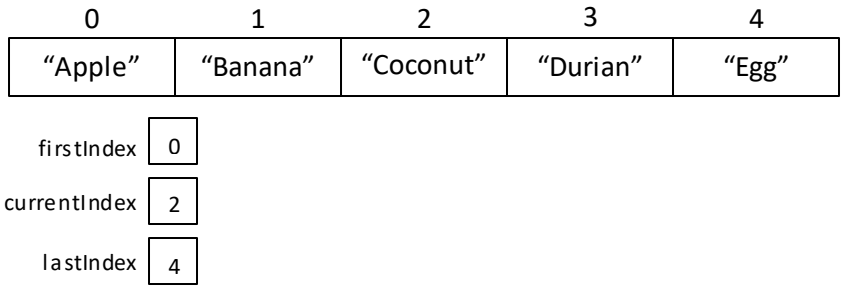


After invoking advance() again:

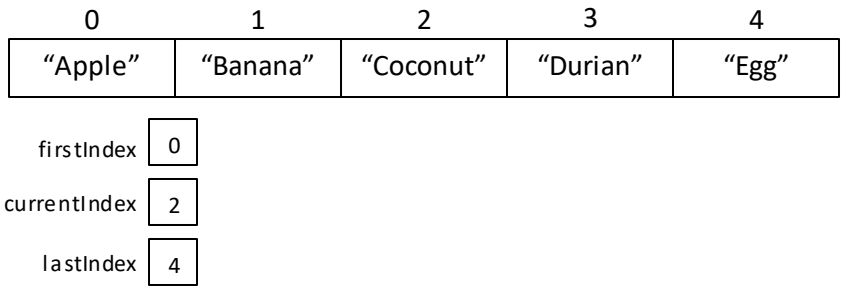


Invoking getCurrentEntry() will return "Coconut".

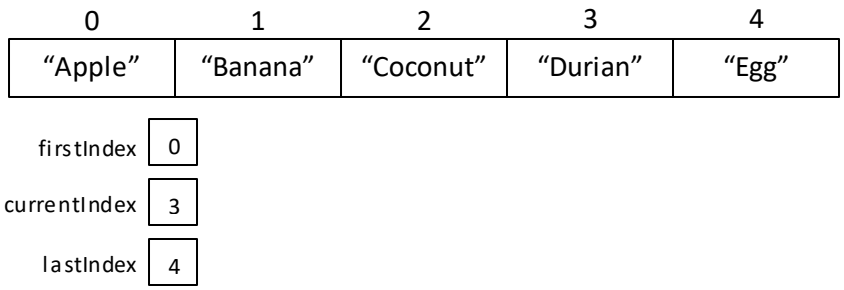
After invoking add("Egg"):



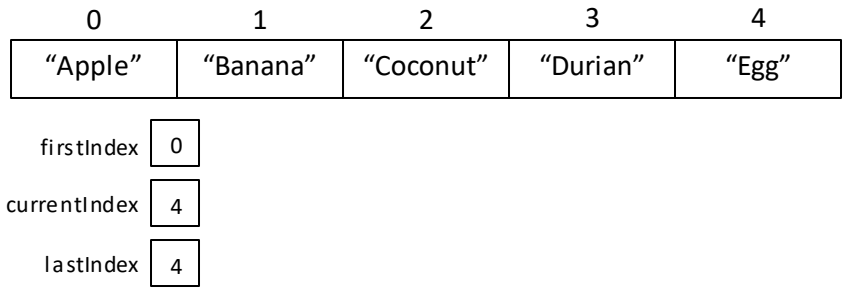
After invoking add("Fig"), you should get "Cannot add. No more empty slot in the ring."



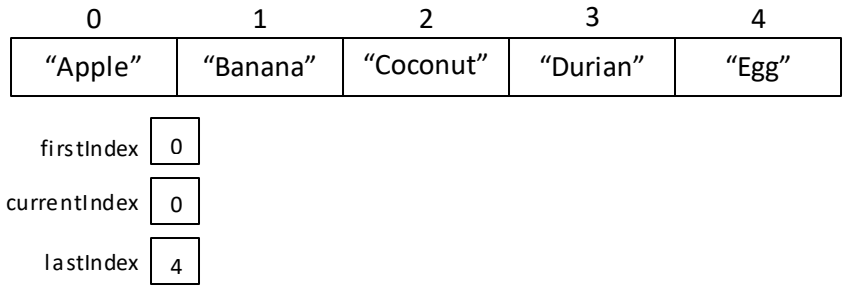
After invoking advance():



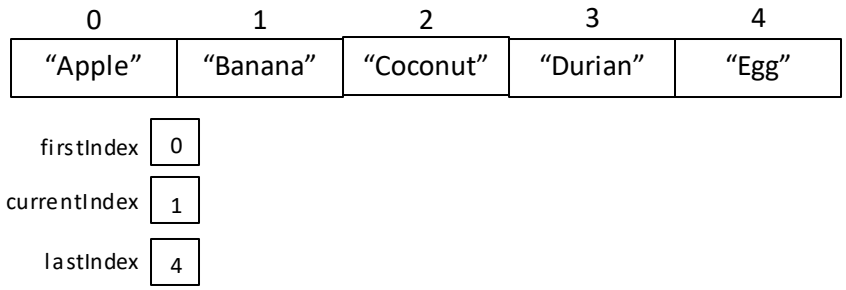
After invoking advance() again:



After invoking advance() again:



After invoking advance() again:



Invoking getCurrentEntry() will return "Banana".

b) Define a class *TestArrayRing* to test your implementation of *ArrayRing*. The *TestArrayRing* class must be saved in a file named "TestArrayRing.java". Your class should include statements that call all the 6 methods and produce the following output:

```
Create a ring to store 5 String objects
Calling isEmpty() should return true. Actual result: true
Add these 4 String objects to the ring: Apple, Banana, Coconut, Durian.

Calling getLength() should return 4. Actual result: 4 entries

Calling getCurrentEntry() should return Apple as current entry. Actual result: Apple
Calling advance() twice followed by calling getCurrentEntry()
    should return Coconut as current entry. Actual result: Coconut

Add this String object to the ring: Egg
Add this String object to the ring: Fig. You should
    get "Cannot add. No more empty slot in the ring."
Actual Results: Cannot add. No more empty slot in the ring.

Display all entries in the ring starting from current entry which is Coconut.
Coconut is current entry.
Durian is current entry.
Egg is current entry.
Apple is current entry.
Banana is current entry.

Testing remove: remove entry from the ring one by one
Apple is removed from the ring.
Banana is removed from the ring.
Coconut is removed from the ring.
Durian is removed from the ring.
Egg is removed from the ring.

Calling isEmpty() should return true. Actual result: true
Calling getCurrentEntry() should return null. Actual result: null
Calling remove() should return null. Actual result: null

Testing Done.
```

Question 3 (5 marks)

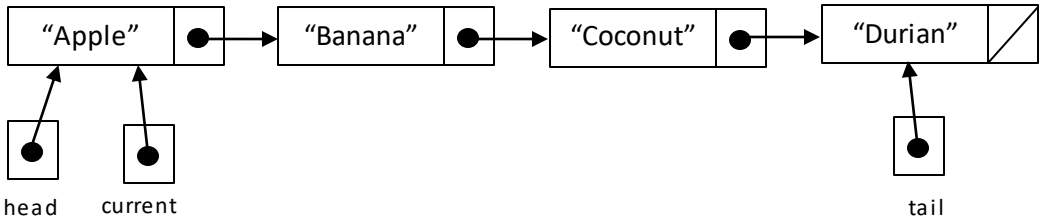
a) Define a class *LinkedRing* that implements your *RingInterface<T>* from Question 1. An object of *LinkedRing* can store maximum 5 entries only. Use linked list in your implementation. The *LinkedRing* class must be saved in a file named "LinkedRing.java". No marks will be given for any implementation using the existing Collection classes available from the Java API Library.

You must have the following data members:

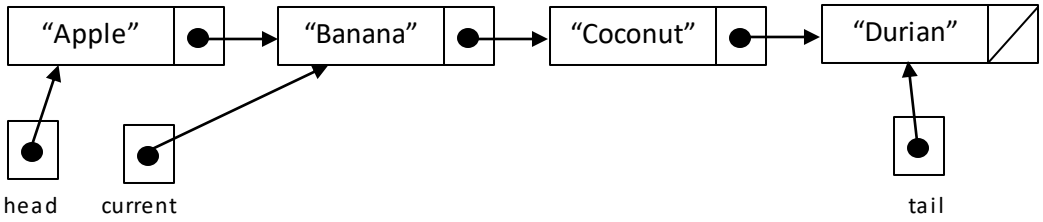
head - Marks the first entry
tail - Marks the last entry
current - Marks the current entry. This is the reference/marker to the current item.

Initially *current* marks the first entry.
Calling the *advance* method will advance *current* by one entry.
Add method will add an item to the end of the ring but cannot add an item if there is no more empty slot in the ring.

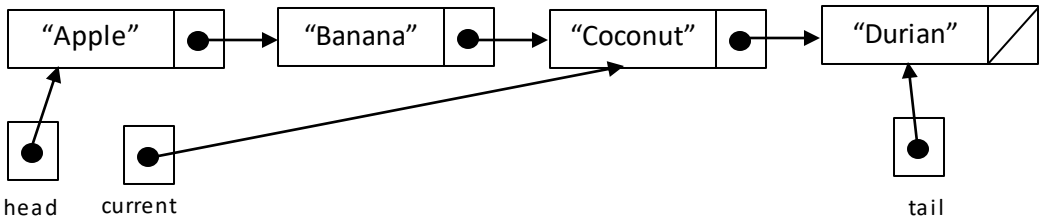
Read part b) especially the required output together with the following illustrations before you start this question. This will help you to get a better understanding.



After invoking advance():

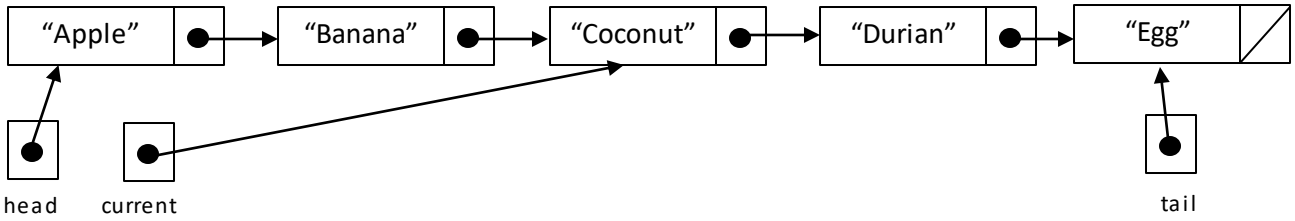


After invoking advance() again:

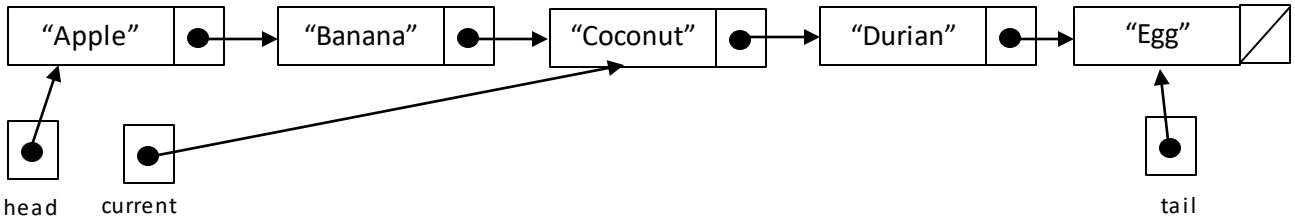


Invoking getCurrentEntry() will return "Coconut".

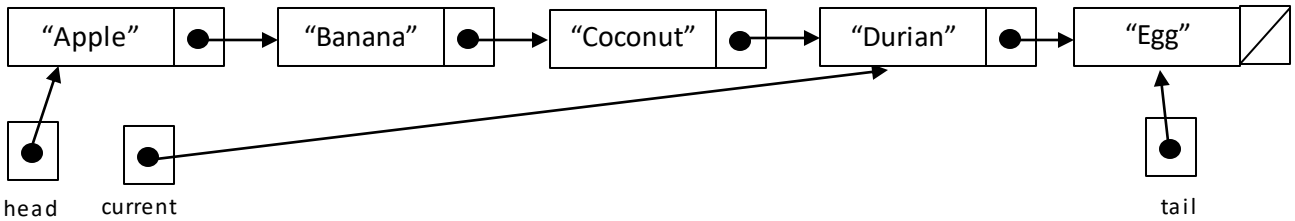
After invoking add("Egg"):



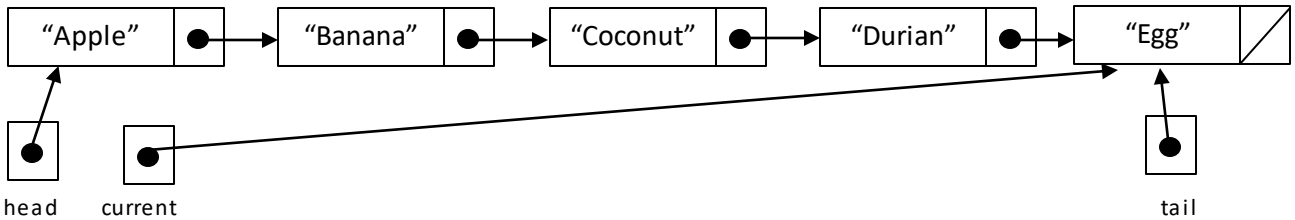
After invoking add("Fig"), you should get "Cannot add. No more empty slot in the ring."



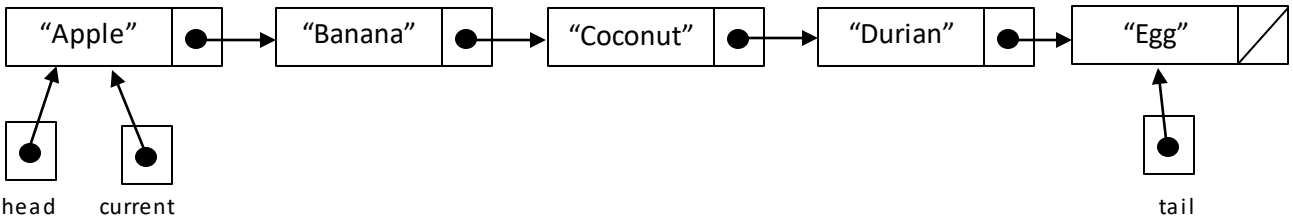
After invoking advance():



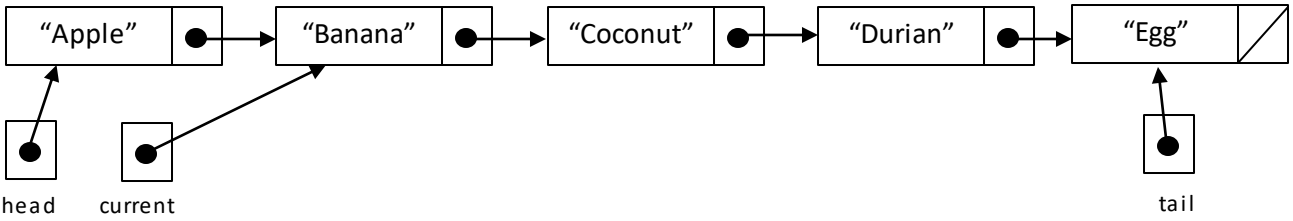
After invoking advance() again:



After invoking advance() again:



After invoking advance() again:



Invoking getCurrentEntry() will return "Banana".

b) Define a class *TestLinkedRing* to test your implementation of *LinkedRing*. The *TestLinkedRing* class must be saved in a file named "TestLinkedRing.java". All the statements **except ONE statement** in *TestLinkedRing* class must be identical to the statements in your *TestArrayRing* class from Question 2.b. When you execute *TestLinkedRing* class, it should produce the same output as shown in Question 2. b.

THE END

The marking scheme for Mid Term Test (15 marks):

Question 1 (5 marks)
Interface – 1 mark
Generic – 1 mark
6 methods – 3 marks

Question 2 (5 marks)
No output, minimal code – 0 mark
No output, some working code – 1 mark
25% required output, 25% working code – 2 marks
50% required output, 50% working code – 3 marks
75% required output, 75% working code – 4 marks
100% required output, 100% working code – 5 marks

Question 3 (5 marks)
Same as Question 2.